Credit Card Fraud Detection Report

Neha Agnihotri
Sejal Chandak
College of Engineering
Northeastern University
Boston, MA
chandak.se@northeastern.edu
agnihotri.n@northeastern.edu

Abstract

In this report, we have explored the credit card fraud detection with an interesting datasets. The three learning algorithms include Naive Bayes , Logistic regression and Random Forest. The learning curve, model complexity and training time of each algorithm on both datasets have been explored and analyzed.

Introduction

Background: As technology advances, the entire globe is moving toward digitalization. Credit card use has skyrocketed because of cashless purchases. The number of credit card accounts increased by 2.6% over the previous year. With the increased use of credit cards comes a rise in credit card fraud. Credit card fraud is a form of identity theft that causes massive personal financial deficits, corporate shortfalls, and national economic losses.

Motivation: With the increased use of digital payment methods, the number of fraudulent transactions is expanding in novel and unexpected ways. Credit card fraud detection utilizing machine learning is not only a trend in the banking sector, but it is also a need for them to have proactive monitoring and fraud protection measures. Machine learning is assisting these institutions in reducing time-consuming manual checks, expensive chargebacks and fees, and valid transaction denials.

Goal: It becomes important to analyze credit card fraud as new data emerges. Understanding such trends using existing machine learning algorithms and models can aid in the creation of Automated Credit Card Fraud Detection Systems. The primary objective of this project is to utilize supervised machine learning-based techniques such as Gaussian Naive Bayes Algorithm, Random Forest and Logistic Regression, and to detect 100% of fraudulent transactions while minimizing the incorrect fraud classifications and to provide a comparative study for performance analysis of the algorithms.

Methodology

Credit card fraud detection necessitates the use of an ML-based algorithm to accomplish feature extraction and model evaluation. In this project, three machine learning techniques: Logistic Regression, Gaussian Naive Bayes, and Random Forest are compared and analysed by utilizing the credit card fraud dataset from Kaggle to create a fraud detection system.

Following is the methodology employed by this project:

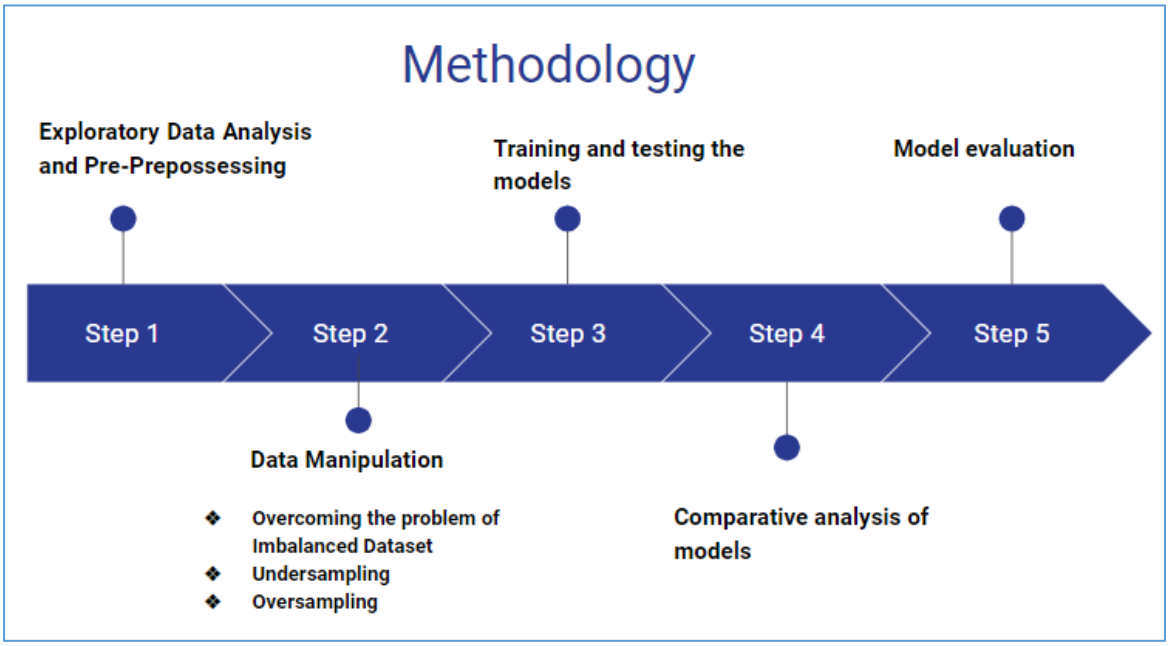


Fig: Methodology

Dataset

Download Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Several datasets were reviewed and analysed for the purpose of fraud detection, and the credit card dataset, which is accessible on Kaggle. This dataset, made by European cardholders in September 2013, comprises 492 frauds out of 284,807 transactions that occurred during a two-day period. The feature 'Class' is the response variable that takes the value of 1 in case of fraud and a value of 0 for non-fraud cases. While exposing a customer's transaction information is considered a confidentiality risk, principal component analysis is employed to examine the bulk of the attributes in the dataset (PCA). V1, V2 to V28 are PCA-applied features, whereas the remaining features, namely 'time,' 'amount,' and 'class,' are non-PCA-applied features. The non-PCA applied features are shown in the table below

Sr no	Feature	Description
1.	Time	Time in seconds between the current ongoing transaction and the first transaction.
2.	Amount	Transaction Amount (in dollars)
3.	Class	Class 0: Non-Fraudulent Class 1: Fraudulent

Table 1: Dataset Description

A. Data Observation:

There is a noticeable statistical imbalance between the data points identified as fraudulent and those labelled as non-fraudulent. The use of this raw data may result in the problem of overfitting. Undersampling, oversampling, and cross-validation can be used to resolve such issues. There are 284315 non-fraudulent transactions and 492 fraudulent ones. In this project, we make use of undersampling and oversampling to tackle the problem of imbalanced data.

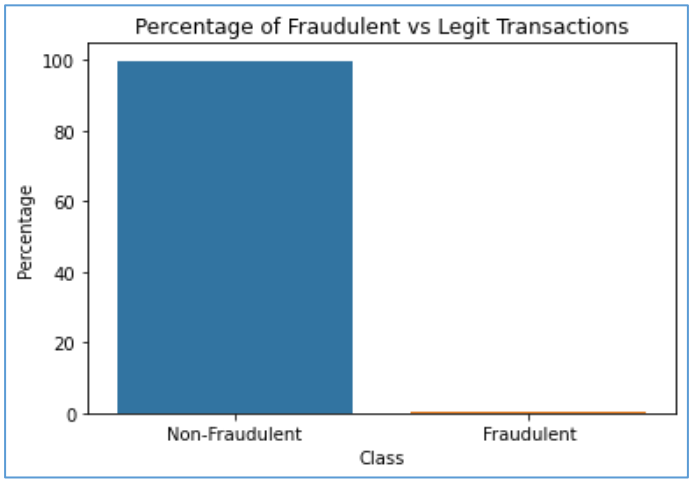


Fig. Class imbalance

B. Exploratory Data Analysis and Pre-Prepossessing:

We used multiple python methods to check the features of the data such as head, describe and info, to understand the data.

```
ccdata.describe()

# The result provide the perfect understanding of count mean std min max for the creditcard_dataset
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	..
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	..
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15	..
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	..
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	..
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	..
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	..
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	..
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	..

8 rows × 31 columns

Fig. Sample of Exploratory Data Analysis

In the next step, we attempt to examine the data to identify significant aspects that will impact the output of our model and eliminate those that are superfluous. We will also standardize all the model's characteristics here.

C. Data Pre-processing

To visually interpret the dataset and check for irregularities, we used the following:

1. <u>Handling Missing Values:</u>

Initially we check for the missing values in the dataset in which we discovered that there were no missing values in the dataset.

Handling Missing Values

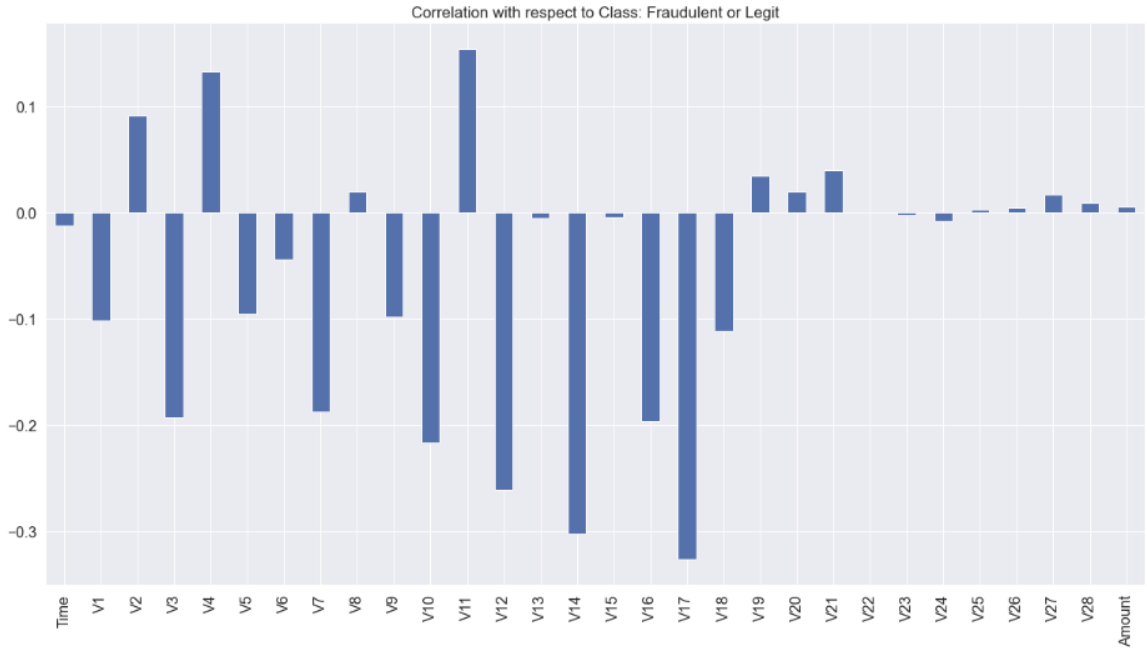
```
In [210]: # Checking the presence of any missing values in the creditcard_dataset

          ccdata.isnull().sum().sum()

Out[210]: 0
```

We understood that there are zero missing values in the entire creditcard_dataset

Fig. Code snippet

2. <u>Correlation with Class</u>

To understand the correlation between the classes and the features we plotted a bar graph and heatmap.



We understand that there are features like v22 has no correlation with the class and can be dropped to optimize the model training.

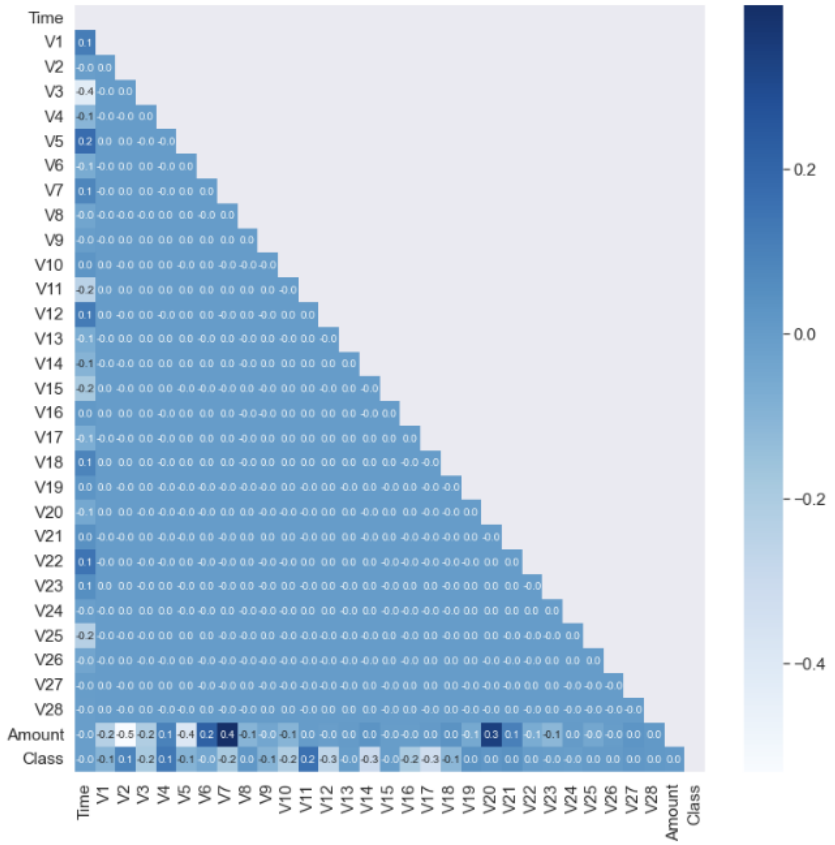


Fig. Feature heatmap

In figure, the heat-map above shows that there are no feature columns with strong correlation value with each other. While there is no predictor column that has a strong correlation value with the Class column, we observe a negative association between V2 and Amount, as well as a positive correlation between V7 and the Amount characteristic.

3. Distribution of Class and Time:

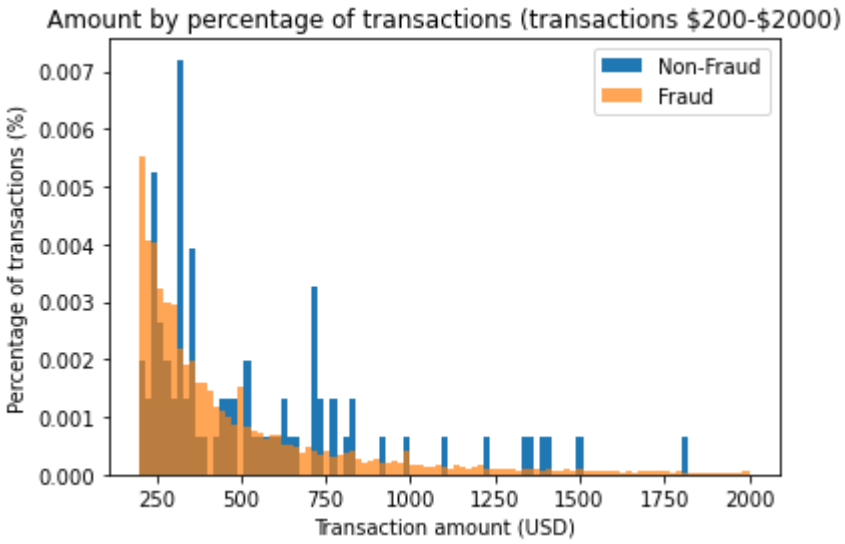


Fig. 3. Transaction amount

We may infer from the above graph of transaction amount (in dollars) and percentage of total transactions made that fraud happened more frequently in the long tail. The transaction amount appears can be used for distinguishing between fraud and non-fraudulent transactions.

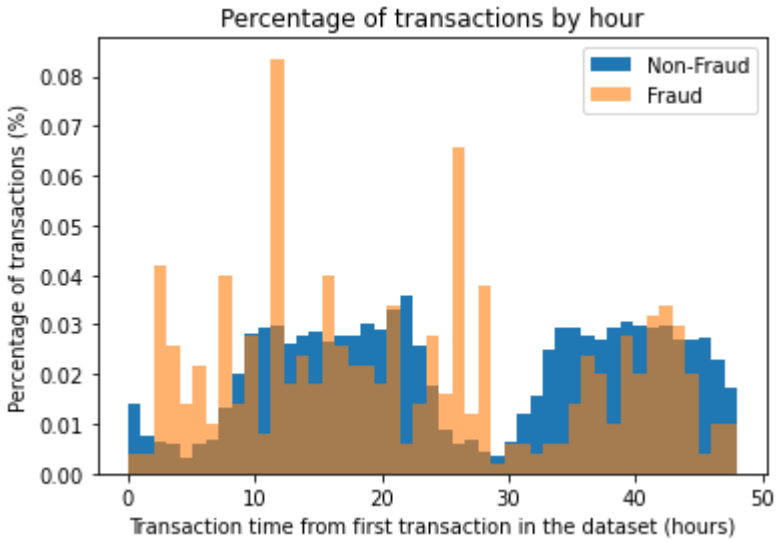


Fig.Transaction time

The graph represents the relationship between transaction time in hours (from the first to the final hour) and the percentage of transactions processed over the course of two days. Given the large drop in non-fraudulent transactions between hours 1 and 8, and again between hours 24 and 32, it indicates that fraud happens more frequently at night.

4. Feature Scaling:

As previously known, principal component analysis is employed to examine the bulk of the attributes in the dataset and altered features V1–V28. Whereas feature "Time" and "Amount" have not. Additionally, they must to be scaled before we train our model using different approaches because we'll be analyzing these two features along with others from V1 to V28.

We have used standard scalar for feature scaling on the amount and time column of the data.

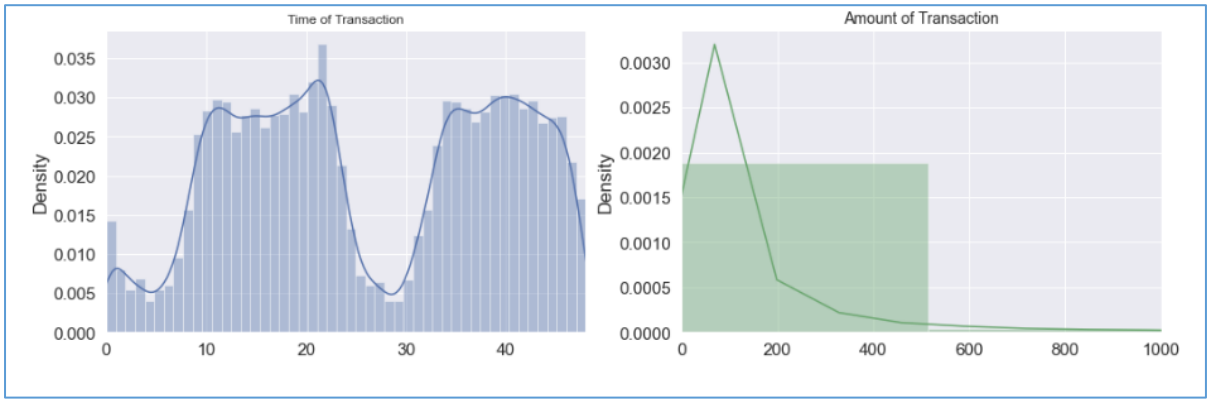


Fig. Data Before Normalizing

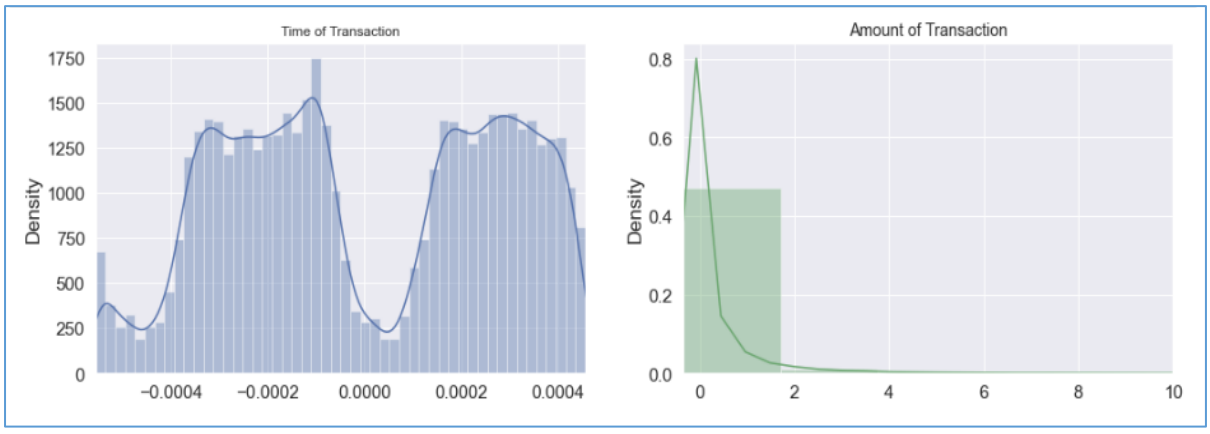


Fig. Data After Normalizing

5. Checking Skewness:

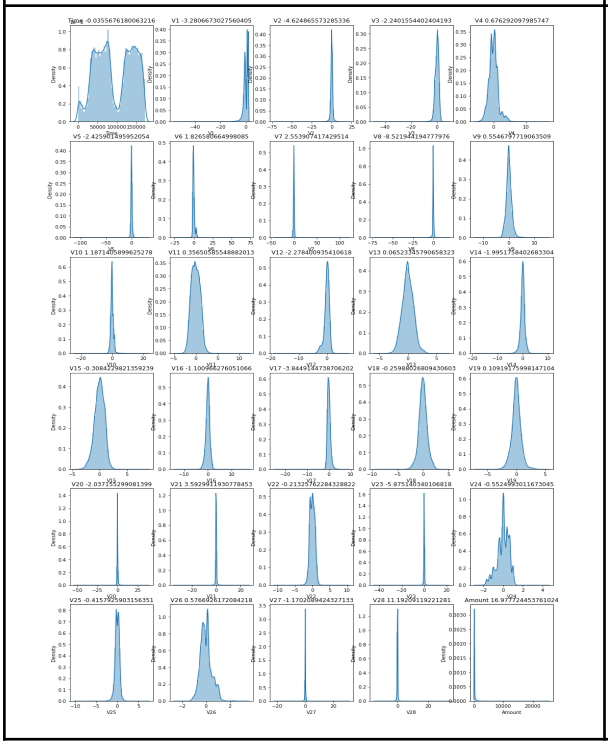
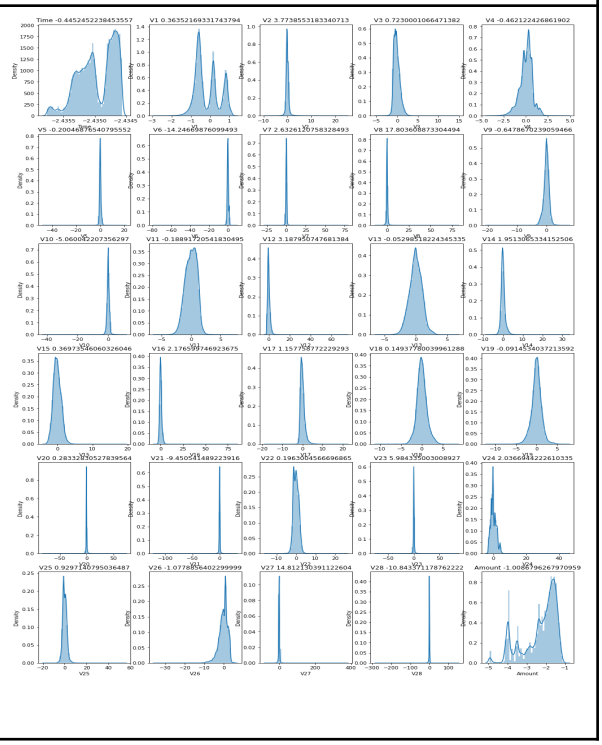
Before applying Power Transformer	After applying Power Transformer
	

Fig. Distribution by Power Transformer

After examining the feature distribution, we observe that all the features are significantly skewed; there is a need to reduce the skewness of the data for a symmetric distribution. To reduce skewness, a data transformation is employed. A symmetric or nearly symmetric distribution is usually easier to handle and understand than a skewed distribution. We use Power Transformer to achieve a uniform distribution of data and to stabilize the distribution's fluctuation. From the above two figures, a comparison of feature distribution is plotted after applying a power transformer.

Overcoming the problem of an imbalanced dataset by using sampling techniques

A balanced sample of data with the same frequency of fraudulent and non-fraudulent transactions is required, such that the dataset has a ratio of the same number of fraudulent and non-fraudulent transactions, allowing algorithms to generate more accurate and exact findings. Here we tackle the problem of an imbalanced dataset by using two methods:

a. Undersampling

A balanced sample of data with the same frequency of fraudulent and non-fraudulent transactions is taken into consideration, such that the dataset has a ratio of around 50/50 fraudulent and normal transactions, allowing algorithms to generate more exact findings.

The reason for utilizing under-sampling in our dataset is to detect fraudulent transactions more accurately, and under sampling has no effect on the fraud data in the original dataset. After under-sampling, we get 492 values for both fraudulent and non-fraudulent transactions; thus, an even distribution is obtained.

b. Oversampling

Oversampling is used where minority class instances are added by replicating training samples with the same class representation. After oversampling, we get 284315 values for both fraudulent and non-fraudulent transactions; thus, an even distribution is obtained.

Training and Testing Data:

Before oversampling and under-sampling, we partition the dataset into training and testing parts, and then only oversample or under sample the training portion. The total quantity of data collected will be divided in 3:1 train data, test data. It denotes that 67% of the total data will be used to train the ML model, with the remaining 33% used to test the model's accuracy.

Training and Testing Machine Learning Models: We used three different machine learning algorithms, trained, and tested them all with the same dataset. These three algorithms are as follows:

- **Logistic regression:** Logistic regression is a widely used Machine Learning method that belongs to the Supervised Learning approach. It is used to predict the categorical dependent variable from a group of independent factors. As a result, the conclusion must be categorical or discrete. It can be No or Yes, 0 or 1, True or False, etc., but instead of giving the precise values like 0 and 1, it delivers the probability values that fall between 0 and 1, allowing it to classify whether the transaction is fraudulent or non-fraudulent

- **Random forest:** Random Forest is a well-known machine learning algorithm from the supervised learning approach. It may be applied to both classification and regression issues in machine learning. Random Forest is a supervised machine learning technique that uses ensemble learning to mix many algorithms of the same type, i.e.. multiple decision trees, resulting in a forest of trees, thus the name "Random Forest." When splitting a node, it looks for the best feature from a random subset of characteristics rather than the most essential feature. As a result, there is a greater variety, which leads to a better model. We utilize the algorithm for credit card fraud detection since it is not biased and offers advantages such as increased dimensionality and accuracy.

- **Naive Bayes:**

  A Naive Bayes classifier is a type of probabilistic machine learning model used for classification tasks. It is based on the Bayes theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

From the above formula, given that B occurred, we may compute the probability of A happening. In this situation, B stands for evidence, and A stands for hypothesis. In this situation, the traits are believed to be independent of each other. In other words, the presence of one trait has no bearing on the presence of another. Naive Bayes models are a group of extremely fast and simple classification algorithms and become a base line for classification problems. We have used Gaussian Naïve Bayes algorithm as the assumption is that data from each label is drawn from a simple Gaussian distribution.

Result and Analysis

a. Performance Metrics and Experimental Results:

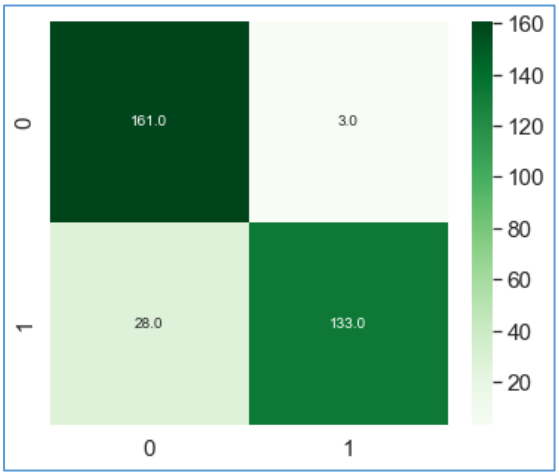
When working with excessively imbalanced data, accuracy is not always valid performance metric. A credit card company wants to eliminate false negatives as much as possible since fraudulent transactions may be extremely costly, and a false alert suggests that someone's transaction is prohibited (reduce false positives). As a result, the credit card company wants to increase recalls. F1-score considers a balance between precision and recall. A Confusion matrix is a 2 x 2 matrix used for evaluating the performance of a classification model. The matrix compares the actual target values with those predicted by the machine learning model. Confusion matrix uses performance metrics like accuracy, precision, recall, and F1-score. For evaluating the performance of three models we take confusion matrix, Classification report and AUC ROC graph into consideration for determining the best model.

Performance Metrics

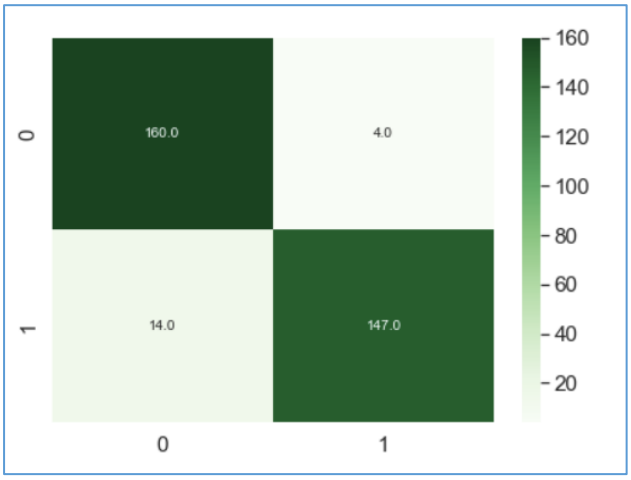
1. **Confusion Matrix:**

UNDERSAMPLING RESULTS:

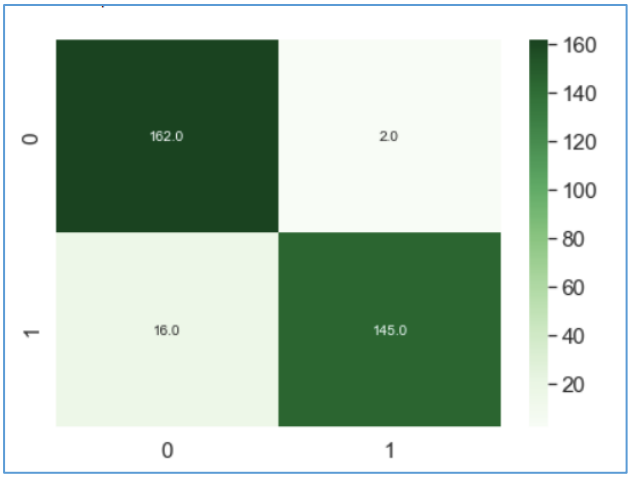
A. Naive Bayes:



B. Logistic Regression:

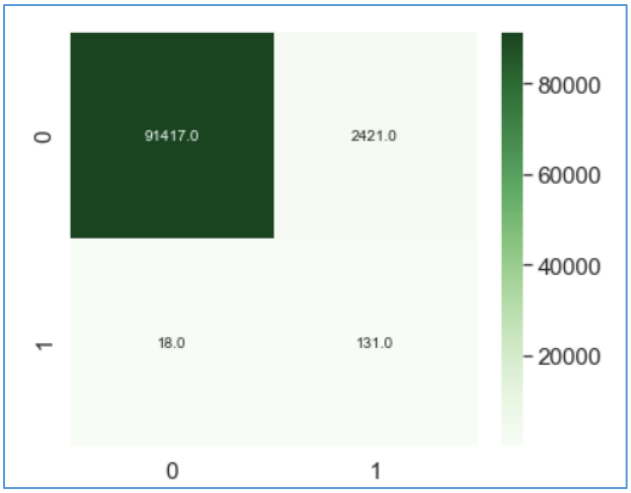


C. Random Forest:

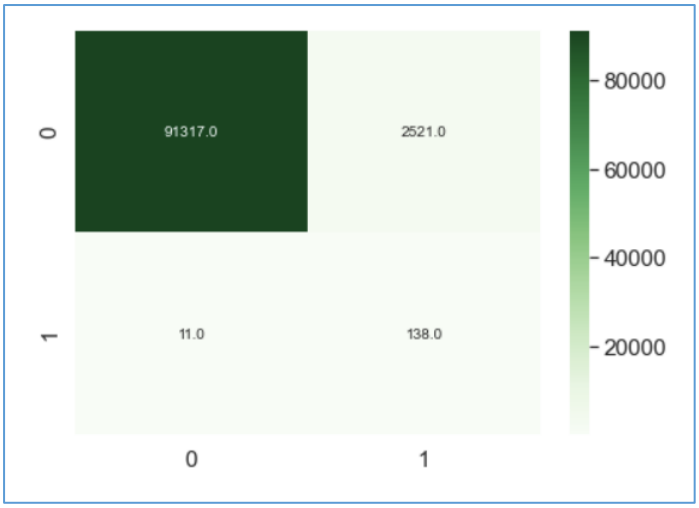


OVERSAMPLING RESULTS:

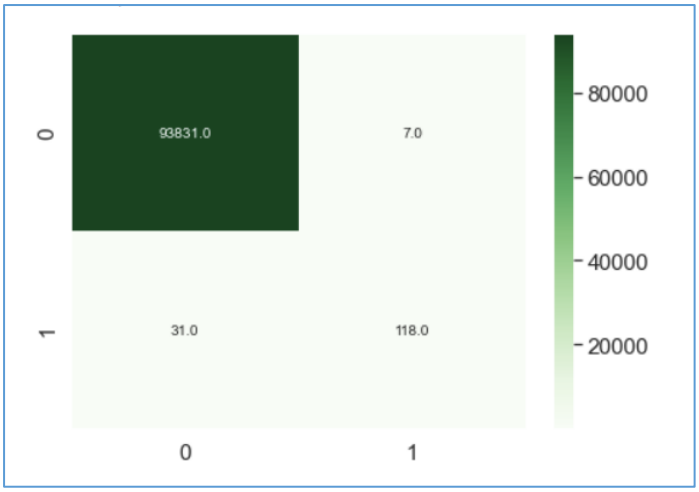
A. Naive Bayes:



B. Logistic Regression:



C. Random Forest:



2. **Classification Report and ROC**

UNDERSAMPLING RESULTS:

A. Naive Bayes: Naive Bayes model - being the first model gave an accuracy of 90% with AUC-ROC value of 0.96 and cross validation score of 92%

Testing				
Class	Precision	Recall	F1-Score	Support
0	0.85	0.98	0.91	164
1	0.98	0.83	0.90	161

Fig. Classification Report for Naïve Bayes

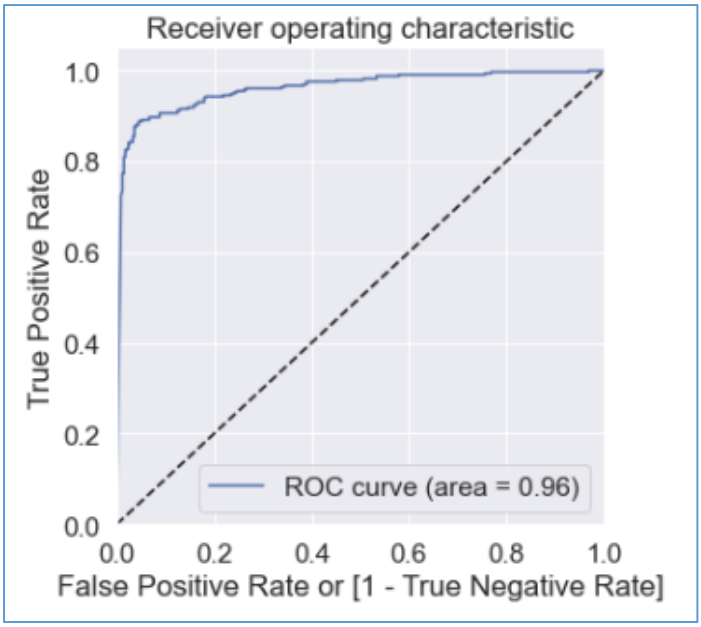


Fig. AUC-ROC for Naïve Bayes

B. Logistic Regression: Logistic regression performs well with our dataset, accuracy of 94%. And cross validation score of 94% and AUC-ROC of 0.97

Testing				
Class	Precision	Recall	F1-Score	Support
0	0.92	0.98	0.95	164
1	0.97	0.91	0.94	161

Fig. Classification Report for Logistic Regression

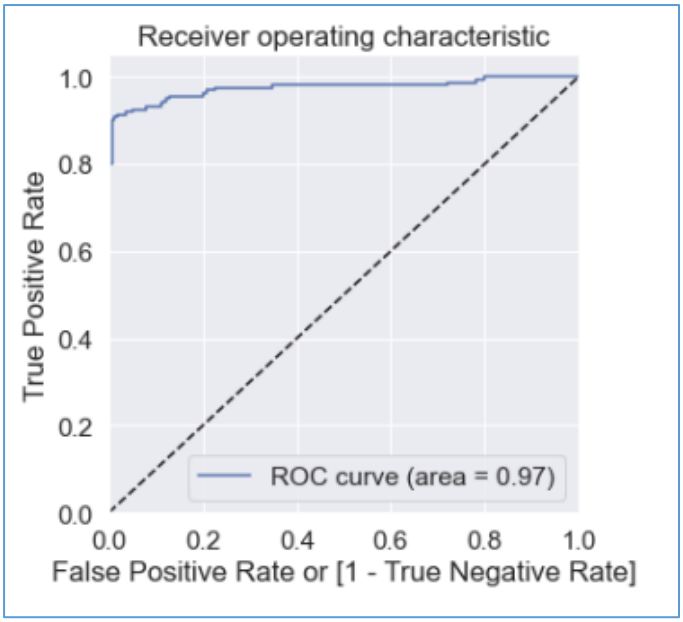


Fig. AUC-ROC for Logistic Regression

C. Random Forest

After comparing the classification reports for training and testing the dataset by using Random Forest, we can conclude that the model has been trained well. The accuracy for Random Forest stands at of 94%, cross validation of 94% and and AUC-ROC of 0.98

Testing				
Class	Precision	Recall	F1-Score	Support
0	0.91	0.99	0.95	164
1	0.99	0.90	0.94	161

Fig. Classification Report for Random Forest

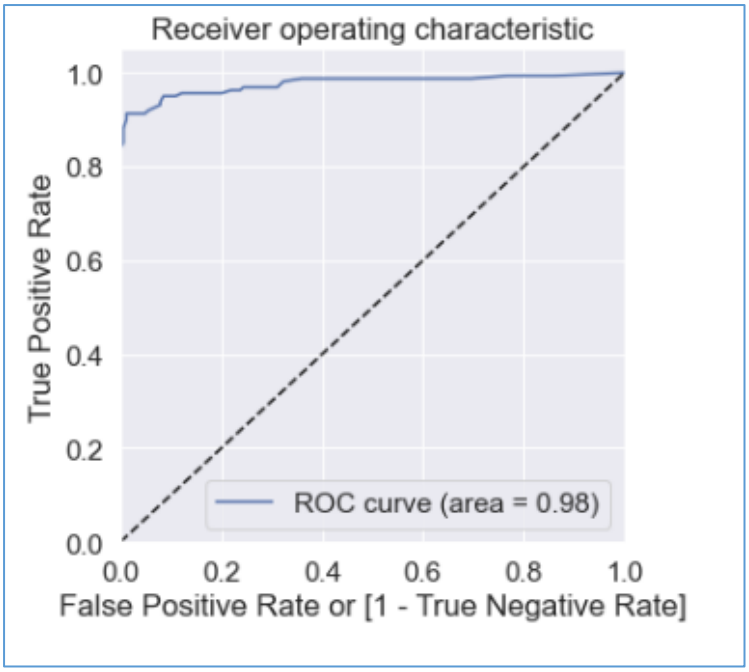


Fig. Classification Report for Random Forest

OVERSAMPLING RESULTS:

Performance Metrics: Classification Report and ROC

A. Naive Bayes: The Naive Bayes model - being the first model, gave accuracy of 97% with an AUC-ROC value of 0.97 and a cross validation score of 91%

Testing				
Class	Precision	Recall	F1-Score	Support
0	1.00	0.97	0.99	93383
1	0.05	0.88	0.10	149

Fig. Classification Report for Naïve Bayes

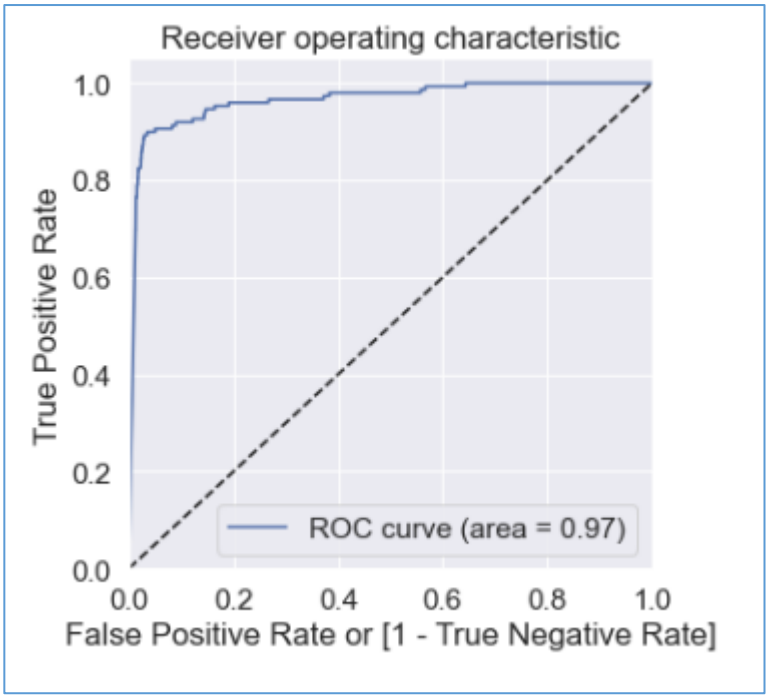


Fig. AUC ROC for Naïve Bayes

B. Logistic Regression: Logistic regression performs well with our dataset, with a accuracy of 97%. And cross validation score of 95% and AUC-ROC of 0.98

Testing				
Class	Precision	Recall	F1-Score	Support
0	1.00	0.97	0.99	93383
1	0.05	0.93	0.10	149

Fig. Classification Report for Logistic Regression

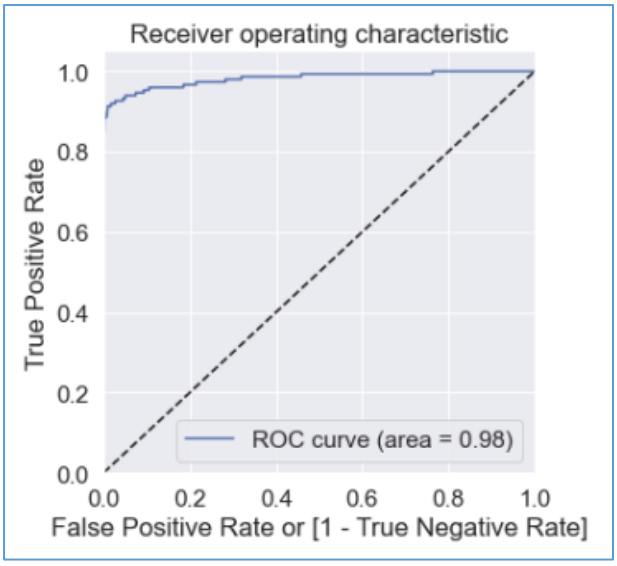


Fig. AUC ROC for Logistic Regression

C. Random Forest

After comparing the classification reports for training and testing the dataset by using Random Forest, we can conclude that the model has been trained well. The accuracy score of 100%, cross validation of 100% and AUC- ROC of 0.99

Testing				
Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	93383
1	0.94	0.79	0.86	149

Fig. Classification Report for Random Forest

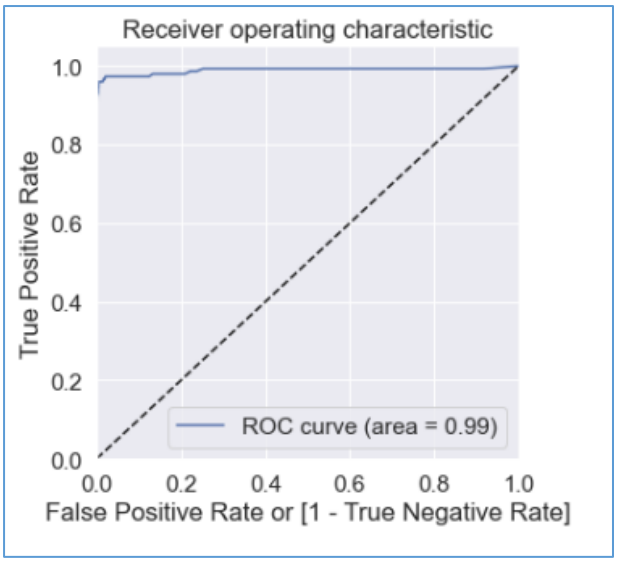


Fig. AUC ROC for Random Forest

Conclusion

This project compares and contrasts three models: Gaussian Naive Bayes, Logistic Regression, and Random Forest, all of which were trained and evaluated on the Credit Card Fraud dataset using default parameters using Under sampling and Oversampling strategies. To evaluate each model, standard metrics such as precision, AUC, F1-score, recall, precision were generated. For the oversampling technique, we see an interesting trend for precision, recall, F1 with AUC-ROC score. For Naïve Bayes and Logistic regression model, we see a High AUC and low F1 Score which means the model is reliably assigning higher scores to positive examples than to negative examples. Random Forest outperformed the other models in terms of AUC, F1-score, Recall, with and AUC equal to 0.99 for both under sampling and oversampling.

Future Scope:

Although this approach employs supervised learning, unsupervised learning is an intriguing field to explore. This work might be enhanced further by training the hyper parameters and fine-tuning the model for greater accuracy

Dataset Download Link:

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

References:

[1] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

[2] https://www.solver.com/xlminer/help/neural-networks-classification-intro#:~:text=Artificial%20neural%20networks%20are%20relatively,actual%20classification%20of%20the%20record

[3] https://en.wikipedia.org/wiki/Logistic_regression

[4] https://en.wikipedia.org/wiki/Random_forest

[5] Shivam Gupta, “Machine Learning Approach for Credit Card Fraud Detection (KNN & Naïve Bayes)”

[6] Shen, A., Tong, R., & Deng, Y. (2007). Application of Classification Models on Credit Card Fraud Detection. 2007 International Conference on Service Systems and Service Management.

[7] Muaz, A., Jayabalan, M., & Thiruchelvam, V. (2020). A Comparison of Data Sampling Techniques for Credit Card Fraud Detection. International Journal of Advanced Computer Science and Applications, 11(6). https://doi.org/10.14569/ijacsa.2020.0110660

[8] Gupta, A., Lohani, M. C., & Manchanda, M. (2021). Financial fraud detection using naive bayes algorithm in highly imbalance data set. *Journal of Discrete Mathematical Sciences and Cryptography*, *24*(5), 1559–1572. https://doi.org/10.1080/09720529.2021.1969733

[9] Jake VanderPlas, Python Data Science Handbook.